



```

  LL      IIIII
  LL      IIIII
  LL      II
  LL      II
  LL      II
  LL      II
  LL      II
  LL      II
  LL      II
  LL      II
  LL      II
  LL      II
  LL      II
  LLLLLLLLLL  IIIII
  LLLLLLLLLL  IIIII

  SSSSSSSS
  SSSSSSSS
  SS
  SS
  SS
  SS
  SSSSSS
  SSSSSS
  SS
  SS
  SS
  SS
  SSSSSSSS
  SSSSSSSS
  
```



Facility: SHUTDOWN, Procedure for Performing an Orderly System Shutdown

Abstract: This procedure is invoked to perform an orderly shutdown of a VAX/VMS system. It supports a variety of options, as described below. All necessary steps are performed to clean up the system and bring it down for rebooting.

P1 Number of minutes until final shutdown.  
P2 Reason for shutdown.  
P3 Should the disk volumes be spun down?  
P4 Should SYSHUTDOWN.COM be invoked?  
P5 Time when system will be rebooted.  
P6 Should system be rebooted automatically?  
P7 Comma-separated list of keywords. Legal keywords: REBOOT\_CHECK, CLUSTER\_SHUTDOWN, REMOVE\_NODE, and NONE

Environment: DCL with the following privileges:

CMKRNL For system utilities.  
EXQUOTA Just in case.  
LOG\_IO For SET TIME.  
OPER For broadcasting messages and SET TIME.  
SYSNAM For manipulating system logical names.  
SYSPRV For system utilities.  
WORLD For stopping all user processes.

Author: Paul C. Anagnostopoulos  
Creation: 8 April 1983

Modifications:

|         |                                                                                                                                                                                                                                         |                   |             |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------|-------------|
| V03-016 | BLS0349                                                                                                                                                                                                                                 | Benn Schreiber    | 30-AUG-1984 |
|         | Uppcase logicals in f\$trnlrm calls.                                                                                                                                                                                                    |                   |             |
| V03-015 | BLS0343                                                                                                                                                                                                                                 | Benn Schreiber    | 25-AUG-1984 |
|         | Do not allow REMOVE_NODE and CLUSTER_SHUTDOWN at the same time. Ensure quorum disk exists.                                                                                                                                              |                   |             |
| V03-014 | BLS0341                                                                                                                                                                                                                                 | Benn Schreiber    | 17-AUG-1984 |
|         | Correct locating of ERRFMT process.                                                                                                                                                                                                     |                   |             |
| V03-013 | BLS0337                                                                                                                                                                                                                                 | Benn Schreiber    | 8-AUG-1984  |
|         | If SHUTDOWN\$INFORM_NODES logical is defined, only reply to those nodes. (\$ define shutdown\$inform_nodes 'nod1,nod2')<br>Change message text to 'please log off node <nodename>'.<br>Add executive mode to translations of sys\$node. |                   |             |
| V03-012 | BLS0330                                                                                                                                                                                                                                 | Benn Schreiber    | 3-JUL-1984  |
|         | Don't stop ERRFMT if it is not running.                                                                                                                                                                                                 |                   |             |
| V03-011 | BLS0326                                                                                                                                                                                                                                 | Benn Schreiber    | 25-JUN-1984 |
|         | Correct handling of abbreviations of P7.                                                                                                                                                                                                |                   |             |
| V03-010 | KDM0104                                                                                                                                                                                                                                 | Kathleen D. Morse | 22-May-1984 |
|         | Move SET TIME out of parameter request loop. Add OPER and LOG_IO privileges, in order to do the SET TIME.                                                                                                                               |                   |             |
| V03-009 | KDM0103                                                                                                                                                                                                                                 | Kathleen D. Morse | 18-May-1984 |
|         | Add a couple SET TIME commands so that the MicroVAX I last known system time can be used upon reboot if                                                                                                                                 |                   |             |

TIMEPROMPTWAIT is set to 0.

- V03-011 CWH3011 CW Hobbs 11-MAY-1984  
Don't reply to the whole cluster all the time, only send the first and the last three messages. Change DISMOUNT command to use the /ABORT qualifier.
- V03-010 BLS0312 Benn Schreiber 3-MAY-1984  
Correct handling of P7 abbreviations. Use SYSSSPECIFIC for shutdown.tmp. Dismount library disk if tailored system and library disk is mounted.
- V03-009 BLS0304 Benn Schreiber 16-APR-1984  
Change to DISK\_QUORUM parameter.
- V03-008 BLS0301 Benn Schreiber 11-APR-1984  
Ask about auto-reboot before 'when back up' and adjust default answer if auto-reboot.
- V03-007 BLS0290 Benn Schreiber 22-MAR-1984  
Correct quorum disk handling if clustermember but no quorum disk. Define opcsnodump for opccrash.
- V03-006 BLS0274 Benn Schreiber 22-FEB-1984  
Modify P7 usage, add cluster shutdown features. Don't dismount the quorum disk.
- V03-005 BLS0253 Benn Schreiber 11-Dec-1983  
Make reboot checking controlled by P7.
- V03-004 BLS0248 Benn Schreiber 5-Dec-1983  
Speed up loops, and use INSTALL PURGE command. Add checking to ensure that files needed to reboot the system are present.
- V03-003 CWH3003 CW Hobbs 21-Sep-1983  
Change REPLY command to REPLY /SHUTDOWN so that users will have specific control over receiving shutdown messages.
- V03-002 PCA1026 Paul C. Anagnostopoulos 19-Aug-1983  
Fix a bug in the code that sets up SHUTDOWN\$TIME.
- V03-001 PCA1026 Paul C. Anagnostopoulos 17-Aug-1983  
Tweak the algorithm for dismounting disk volumes so that it is more robust.

If this is a recursive invocation, then we have a callback to perform some function. P1 is the requested function.

```
if f$type(shutdown$in_progress) .nes. "" then goto 'p1
goto begin_shutdown
```



This routine can be recursively invoked to ask the user a question and obtain an answer.

P1 ASK  
 P2 A global symbol to be equated to the answer.  
 P3 The prompt string, with no trailing punctuation.  
 P4 The default answer.  
 P5 B for boolean answer, I for integer, S for string.

```

$ASK:
$ on control y then exit %x10360004
$ on error then exit %x10360004
$ if p4 .nes. "" then p3 = p3 + " [" + p4 + "]"
$ p3 = p3 + f$ext(f$loc(p5, "BIS"), 1, "?:") + " "
$a10: read/end_of_file=a10/prompt=""'p3' sys$command ans
$ if ans .eqs. "" then ans = p4
$ if ans .eqs. "" then goto a10
$ ans = f$edit(ans, "COMPRESS, TRIM") - ....._....._....._.....
$ goto a_p5
$a_B: ans = f$edit(ans, "UPCASE")
$ 'p2 == 1
$ if f$loc(ans, "YES") .eq. 0 then exit %x10360001
$ 'p2 == 0
$ if f$loc(ans, "NO") .eq. 0 then exit %x10360001
$ say "%SHUTDOWN-E-YESNO, Please enter YES or NO."
$ goto a10
$a_I: 'p2 == f$int(ans)
$ if f$type(ans) .eqs. "INTEGER" then exit %x10360001
$ say "%SHUTDOWN-E-INTEGER, Please enter an integer number."
$ goto a10
$a_S: 'p2 == ans
$ exit %x10360001

```

C 7

This routine can be recursively invoked to check a boolean answer for validity and report an error message if it's not.

P1 CHECK\_BOOLEAN  
P2 Answer to be checked.  
P3 Error message.

CHECK\_BOOLEAN:

on control y then exit %x10360004

on error then exit %x10360004

if p2 .eqs. "1" .or. -  
f\$loc(p2,"YES") .eq. 0 .or. f\$loc(p2,"TRUE") .eq. 0 .or. -

p2 .eqs. "0" .or. -

f\$loc(p2,"NO") .eq. 0 .or. f\$loc(p2,"FALSE") .eq. 0 then exit %x10360001

say p3

exit %x10360000



```

$begin_shutdown:
$ shutdown$in_progress = 1
$ say = "write sys$output"
$ recurse = "a" + f$environment('PROCEDURE')
$!
$! Say hello.
$!
$! say f$fa0('!/_/_/_SHUTDOWN -- Perform an Orderly System Shutdown!/_')
$!
$! Remember the current environment and establish the one we need.
$!
$! privs = "CMKRNL, EXQUOTA, OPER, SYSNAM, SYSPRV, WORLD, LOG_IO"
$! saved_privs = f$setprv(privs)
$! if .not. f$privilege(privs) then say "%SHUTDOWN-F-NOPRIV, The following privileges are required:"
$! if .not. f$privilege(privs) then say "-SHUTDOWN-F-NOPRIV, 'privs'"
$! if .not. f$privilege(privs) then exit %x10360004
$!
$! Write system time back to system parameter file for MicroVAX I.
$!
$! SET TIME
$! saved_msg = f$environment('MESSAGE')
$! set message/facility/severity/identification/text
$! saved_uic = f$user()
$! set uic [1,4]
$!
$! Set up a pretty system name.
$!
$! cluster_member = f$getsyi('CLUSTER_MEMBER')
$! system = f$getsyi('NODENAME')
$! if system .eqs. "" then system = f$trnlnm('SYS$NODE', 'LNM$SYSTEM_TABLE', 'EXECUTIVE') - "_ " - " " - ":"
$! nodename = system
$! if system .eqs. "" then nodename = ""
$! if system .eqs. "" then system = "The system"
$!
$! Establish CTRL/Y and error handlers.
$!
$! on control_y then goto CANCELLED
$! on error then goto CANCELLED
$!
$! Get valid values for all of the parameters. Prompt for the ones
$! that were not supplied on the command line.
$!
$! min = f$int(f$trnlnm('SHUTDOWN$MINIMUM_MINUTES', 'LNM$SYSTEM_TABLE'))
$! if f$edit(p1, 'UPCASE') .eqs. 'MINIMUM' then p1 = min
$! if p1 .eqs. "" then recurse ASK shutdown$ -
$! "How many minutes until final shutdown" 'min i
$! if p1 .eqs. "" then p1 = shutdown$
$! if f$type(p1) .nes. 'INTEGER' then say -
$! "%SHUTDOWN-E-INVMIN, Specify an integer for minutes until shutdown."
$! if f$type(p1) .nes. 'INTEGER' then goto CANCELLED
$! p1 = f$int(p1)
$! if p1 .lt. min then say "%SHUTDOWN-E-MINMIN, Minutes until shutdown must be at least 'min.'"
$! if p1 .lt. min then goto CANCELLED
$!
$! if p2 .eqs. "" then recurse ASK shutdown$ -
$! "Reason for shutdown" 'Standalone' s
$! if p2 .eqs. "" then p2 = shutdown$
$!
$! if p3 .eqs. "" then recurse ASK shutdown$ -
$! "Do you want to spin down the disk volumes" 'NO' b
$! if p3 .eqs. "" then p3 = shutdown$
$! recurse CHECK_BOOLEAN 'p3' "%SHUTDOWN-E-INVSPIN, Specify YES or NO for spinning down the disks."
$! if .not. $status then goto CANCELLED

```

```

$
$ if p4 .eqs. "" then recurse ASK shutdown$ -
$   "Do you want to invoke the site-specific shutdown procedure" "YES" b
$ if p4 .eqs. "" then p4 = shutdown$
$ recurse CHECK_BOOLEAN "" p4 "" %SHUTDOWN-E-INVPROC, Specify YES or NO for site-specific shutdown."
$ if .not. $status then goto CANCELLED
$
$ if p6 .eqs. "" then recurse ASK shutdown$ -
$   "Should an automatic system reboot be performed" "NO" b
$ if p6 .eqs. "" then p6 = shutdown$
$ recurse CHECK_BOOLEAN "" p6 "" %SHUTDOWN-E-INVREBOOT, Specify YES or NO for immediate reboot."
$ if .not. $status then goto CANCELLED
$
$ w = "later"
$ if p6 then w = "shortly via automatic reboot"
$ if p5 .eqs. "" then recurse ASK shutdown$ -
$   "When will the system be rebooted" "" w s
$ if p5 .eqs. "" then p5 = shutdown$
$
$ Parse P7
$
$ cmd_p7 = p7
$ get_p7:
$   reboot_check_f = "N"
$   remove_node_f = "N"
$   cluster_shutdown_f = "N"
$   if p7 .nes. "" then goto parse_p7
$   say f$ao("Shutdown options (enter as a comma-separated list):")
$   if cluster_member then -
$     say f$ao("!20<REMOVE_NODE!>Remaining nodes in the cluster should adjust quorum")
$   if cluster_member then -
$     say f$ao("!20<CLUSTER_SHUTDOWN!>Entire cluster is shutting down")
$   say f$ao("!20<REBOOT_CHECK!>Check existence of basic system files")
$   say ""
$   recurse ASK shutdown$ "Shutdown options" "NONE" s
$   p7 = shutdown$
$ parse_p7:
$   if p7 .eqs. "" then goto ask_r_check
$   p7_flags = "/REBOOT_CHECK"
$   if cluster_member then p7_flags = p7_flags + "/REMOVE_NODE/CLUSTER_SHUTDOWN"
$   p7_flags = p7_flags + "/NONE/"
$   tp7 = f$edit(p7,"TRIM,UPCASE") + ","
$ p7_loop:
$   t1 = f$elem(0,"",tp7)
$   if t1 .eqs. "" then goto p7_loopend
$   if t1 .eqs. "NONE" then goto p7_none
$   t2 = t1
$   if f$ext(0,2,t1) .eqs. "NO" then t2 = f$ext(2,999,t1)
$   if t1 .eqs. "RE" and cluster_member then goto bad_p7 !ambiguous
$   t3 = f$loc("/",t1,p7_flags)
$   if t3 .ne. f$length(p7_flags) then goto p7_found
$   t3 = f$loc("/",t2,p7_flags)
$   if t3 .eq. f$length(p7_flags) then goto bad_p7
$ p7_found:
$   t2 = f$element(0,"/",f$extract(t3+1,999,p7_flags))
$   t2_f = (f$locate(t1,t2) .eq. 0)
$   tp7 = tp7 - t1
$   goto p7_loop
$ p7_none:
$   reboot_check_f = "N"
$   remove_node_f = "N"
$   cluster_shutdown_f = "N"
$   tp7 = tp7 - t1

```



```

$ goto p7_loop
$! Here if invalid parameter entered for P7
$bad_p7:
$ say '%SHUTDOWN-E-INVALID_P7, 't1' is an invalid value for the P7 parameter'
$ goto p7_ckredo
$ p7 = ''
$ if cmd_p7 .nes. '' then goto cancelled
$ say
$ goto get_p7
$ p7_loopend:
$ if .not. (remove_node f .and. cluster_shutdown f) then goto ckf_check
$ say '%SHUTDOWN-E=INVREQ, You may not specify both REMOVE_NODE and CLUSTER_SHUTDOWN'
$ p7_ckredo:
$ p7 = ''
$ if cmd_p7 .nes. '' then goto cancelled
$ say
$ goto get_p7
$ ckf_check:
$ if .not. reboot_check_f then goto no_ckf
$ say -
$ f$ao('!/SHUTDOWN-I-BOOTCHECK, Performing reboot consistency check...')
$ !*=contiguous,%=append file with cpu name (780/750/730, etc)
$ !Any files requiring the '%' must rely on previous file to get file type
$ flist = 'SYS$SYSTEM:SYSBOOT.EXE*\VMB*\SYSLOAX\SYSINIT\F11BXQP\RMS\SYS\DCL\TTDRIVER\LOGINOUT\STARTUP.COM'
$ flist = flist + 'SYS$LIBRARY:DCLTABLES.EXE\MOUNTSHR\SYS$SYSTEM:SYSGEN\JOBCTL\VMOUNT\
$ if cluster_member then flist = flist + 'SYS$SYSTEM:CLUSTERLOA\SCSLOA\CSP\
$ cpulist = '0/780/750/730/790/8SS/8NN/UV1/UV2/'
$ cpu = f$getsyi('cpu')
$ rlf = ''
$ errors = 0
$ !Loop through the list of files, checking each one
$ ckf_loop:
$ f = f$elem(0,'%',flist)
$ if f .eqs. '' then goto ckf_done
$ fl = f - '%'
$ if f$loc('%',f) .ne. f$len(f) then goto ckf_cpu
$ ckf_cont:
$ tf = f$search(f$parse(fl,,rlf,'SYNTAX_ONLY'))
$ if tf .eqs. '' then goto ckf_fnf
$ if f$loc('*',f) .eq. f$len(f) then goto ckf_next
$ if .not. f$file(tf,'CTG') then goto ckf_fnc
$ ckf_next:
$ if tf .eqs. '' then goto ckf_next80
$ if rlf .eqs. '' then rlf = tf
$ if f$parse(fl,'%',TYPE,'SYNTAX_ONLY') .eqs. '' then goto ckf_next20
$ rlf = rlf - f$parse(rlf,'%',TYPE,'SYNTAX_ONLY')
$ rlf = rlf + f$parse(fl,'%',TYPE,'SYNTAX_ONLY')
$ ckf_next20:
$ i = f$loc(':',f1)
$ if i .eq. f$len(f1) then goto ckf_next40
$ j = f$loc(':',rlf)
$ rlf = rlf - f$ext(0,j,rlf)
$ rlf = f$ext(0,i,f1) + rlf
$ ckf_next40:
$ ckf_next80:
$ flist = flist - '%f\
$ goto ckf_loop
$ ckf_cpu:
$ t = f$elem(cpu,'/',cpulist)
$ if t .eqs. '/' then goto ckf_next
$ fl = fl + t
$ goto ckf_cont
$ ckf_fnf:

```

```

$      if f1 .eqs. "VMB" and. ((cpu .eq. 7) .or. (cpu .eq. 8)) then goto ckf_next
$      tf = f$parse(f1,,rlf)
$      say "%SHUTDOWN-E-OPENIN, Error opening 'tf' as input"
$      errors = errors + 1
$      goto ckf_next
$ckf_fnc:
$      tf = f$parse(f1,,rlf)
$      say "%SHUTDOWN-E-NOTCONTIG, File 'tf' is not contiguous"
$      errors = errors + 1
$      goto ckf_next
$ckf_done:
$      if errors .ne. 0 then goto conchkerrs
$      say "%SHUTDOWN-I-CHECKOK, Basic reboot consistency check completed"
$no_ckf:
$      say ""
$define_shutdown_time:
$!      Establish a system logical name that tells people when the final
$!      shutdown will occur.
$
$      w = f$cvtime(''+f$str(p1/60)+':'+f$str(p1 - p1/60*60),'ABSOLUTE')
$      define /system /exec shutdown$time ""w"
$
$!      Now we are going to begin a loop which passes the time until final
$!      shutdown. The basic idea is to broadcast a message, then divide
$!      the remaining time in half, wait, and broadcast again. Special
$!      processing is performed at the 6-minute mark so that the system
$!      will be running stand-alone.
$
$      remaining = p1
$      stand_alone = 0
$      queues_stopped = 0
$      to_whom = "/all"
$      inform_nodes = f$trnlrm('SHUTDOWN$INFORM_NODES','LNM$PROCESS_TABLE')
$      if inform_nodes .nes. "" then to_whom = '7all/node=('inform_nodes')'
$      log_out = ""
$      halve_table = '0,0,1,2,2,3,3,4'
$30:      if remaining .gt. 6 .or. stand_alone then goto 33
$
$!      The first time we pass the 6-minute mark, we need to
$!      make the system stand alone, by disabling interactive
$!      logins and shutting down the network. We also make this
$!      an operator's terminal so the user can see operator
$!      messages.
$
$      say "%SHUTDOWN-I-OPERATOR, This terminal is now an operator's console."
$      reply/enable
$      say "%SHUTDOWN-I-DISLOGINS, Interactive logins will now be disabled."
$      set login/interactive=0
$      log_out = " Please log off."
$      if nodename .nes. "" then log_out = " Please log off node 'nodename'."
$      w = f$trnlrm('SYS$NODE','LNM$SYSTEM_TABLE','EXECUTIVE') .nes. ""
$      if w then say "%SHUTDOWN-I-SHUTNET, The DECnet network will now be shut down."
$      if w then mcr ncp set executor state shut
$      stand_alone = 1
$33:
$
$!      The first time we pass the 1-minute mark, we want to stop
$!      the queue manager.
$
$      if remaining .gt. 1 .or. queues_stopped then goto 36
$      say "%SHUTDOWN-I-STOPQUEMAN, The queue manager will now be stopped."
$      set noon

```



```

$                                     H 7
$ if f$search('SYSS$SYSTEM:QUEMAN.EXE') .nes. '' then stop/queue/manager
$ set on
$ queues_stopped = 1
$36:
$! Broadcast a shutdown message to all terminals the first time,
$! and to all users from then on. Then determine the next
$! half-time interval and wait until then.
$
$ reply_done = 1
$ if remaining .gt. 2 then goto 37
$ if inform_nodes .eqs. '' then to_whom = to_whom + '/nonode'
$ if inform_nodes .nes. '' then to_whom = to_whom + '/node=('inform_nodes')'
$37:
$ msg = f$fa0('!AS will shut down in !UL minute!%S; back up !AS.!AS!//!AS', -
$       system,remaining,p5,log_out,p2)
$ reply,to_whom/bell/shutdown,,msg
$ say
$ to_whom = '/user/node'
$ if remaining .eq. 0 then goto 39
$ half = f$elem(remaining,'',halve_table)
$ if half .eqs. '' then half = remaining / 2
$ delay = remaining - f$int(half)
$ wait 'f$int(delay/60):'f$int(delay - delay/60*60)
$ remaining = f$int(half)
$ goto 30
$39:
$! Disable error checking from this point on, because we are
$! now committed to the shutdown.
$
$ set noon
$
$! If requested, invoke the site-specific shutdown procedure.
$
$ w = p4 .and. f$search('sys$manager:sysshutdown.com') .nes. ''
$ if w then say '%SHUTDOWN-I-SITESHUT, The site-specific shutdown procedure will now be invoked.'
$ if w then @sys$manager:sysshutdown
$
$! Stop all processes unless they are in system group 0 or 1.
$
$ say '%SHUTDOWN-I-STOPUSER, All user processes will now be stopped.'
$ errfmt = 0
$ mypid = f$getjpi('','PID')
$ context =
$50:
$   pid = f$pid(context)
$   if pid .eqs. '' then goto 59
$   grp = f$getjpi(pid,'GRP')
$   if grp .eq. 1 -
$     .and. f$getjpi(pid,'PRCNAM') .eqs. 'ERRFMT' then errfmt = 1
$   if pid .eqs. mypid .or. grp .le. 1 then goto 50
$   stop/identification='pid'
$   goto 50
$59:
$! Stop the secondary CPU if it exists and hasn't already been stopped.
$
$ if f$search('SYSS$SYSTEM:MP.EXE') .eqs. '' then goto 60
$ define/user sys$output nl:
$ define/user sys$error nl:
$ show cpu
$ secondary = $status
$ if secondary then say '%SHUTDOWN-I-STOPCPU, The secondary processor will now be stopped.'

```

```

$ if secondary then stop/cpu
$! Remove all installed images so deleted ones won't be lost.
$
$60: say "%SHUTDOWN-I-REMOVE, All installed images will now be removed."
$ install = '$install/command_mode'
$ install purge
$! Dismount all mounted volumes, except for the system volume.
$
$ say "%SHUTDOWN-I-DISMOUNT, All volumes will now be dismounted."
$ if .not. f$getsyi('tailored') then goto 61
$ if .not. f$getdvi('LIB$SYSDEVICE','EXISTS') then goto 61
$ if f$getdvi('LIB$SYSDEVICE','MNT') then -
$   @sys$update:vmstailor dismount 'f$getdvi('LIB$SYSDEVICE','FULLDEVNAM')
$61: dmo = 'dismount /nounload'
$ if p3 then dmo = 'dismount /unload'
$ sysdev = f$getdvi('SYSSYSDEVICE','FULLDEVNAM')
$ quorum_disk = ''
$ if cluster member then quorum_disk = f$getsyi('DISK_QUORUM')
$ quorum_disk = f$edit(quorum_disk,'TRIM,COLLAPSE')
$ q_exist = 0
$ if quorum_disk .nes. '' then q_exist = 1
$ if q_exist then q_exist = f$getdvi(quorum_disk,'EXISTS')
$ if .not. q_exist then quorum_disk = ''
$ if quorum_disk .nes. '' then quorum_disk = f$getdvi(f$edit(quorum_disk,'TRIM'),'FULLDEVNAM')
$ define/user sys$output sys$specific:[sysexe]shutdown.tmp
$ show device/mounted
$ open/error=69 shutdown$temp file sys$specific:[sysexe]shutdown.tmp
$62: read/end of file=69/error=69 shutdown$temp file line
$ if f$loc(' ',line) .eq. f$len(line) then goto 62
$ if .not. f$getdvi(f$elem(0,' ',line),'EXISTS') then goto 62
$ dev = f$getdvi(f$elem(0,' ',line),'FULLDEVNAM')
$ if f$getdvi(dev,'DEVCLASS') .ne. 1 .or. -
$   .not. f$getdvi(dev,'MNT') .or. dev .eqs. sysdev then goto 62
$ if dev .eqs. quorum_disk then goto 62
$ say "%SHUTDOWN-I-DISMOUNT:DEV, Dismounting device 'dev.'"
$ dmo /abort 'dev'
$ goto 62
$69: close /nolog shutdown$temp file
$ if f$search('sys$specific:[sysexe]shutdown.tmp') .nes. '' then -
$   delete sys$specific:[sysexe]shutdown.tmp;*
$!
$! Place a final message in the console log and then close it.
$! Disable this terminal as an operator's console so it won't still
$! be one when we reboot. Wait long enough for the stuff to complete.
$
$ if f$search('SYSSYSTEM:REQUEST.EXE') .nes. '' then request -
$   'f$ao('!AS shutdown was requested by the operator.',system)'
$ reply/nolog
$ reply/disable
$ wait 00:00:08
$!
$! Stop the error formatter and then bag the system. Several logical names
$! are defined to control OPCCRASH.
$
$ if errfmt then stop errfmt
$ define opc$unload 'p3
$ define opc$reboot 'p6
$ define opc$cluster_shutdown 'cluster_shutdown_f
$ define opc$remove_node 'remove_node_f

```



define opcsnodump y

Write system time back to system parameter file for MicroVAX I.

SET TIME

run sys\$system:opccrash

say "%SHUTDOWN-F-OPCCRASH, The system could not be crashed."

exit %x10360004

```

K 7
$! Here if the reboot consistency check fails. Tell the user to
$! correct the problem before shutting down the system
$onchkerrs:
$ say "%SHUTDOWN-E-NOREBOOT, The system will be unable to reboot"
$ say "%SHUTDOWN-E-NOREBOOT, Correct above errors before shutting down the system"
$ goto cancelled
$! We come here if the user enters CTRL/Y during the shutdown procedure.
$! Clean up and restore the original environment. Tell the users with
$! the /SHUTDOWN reply, so that everyone who saw the shutdown messages
$! will also see the cancel message.
$
$CANCELLED:
$ if f$type(reply_done) .nes. "" then reply/all/bell/shutdown.-
$      ""f$ao("!AS shutdown has been cancelled.",system)
$
$ close /nolog shutdown$temp file
$ if f$trlnm("SHUTDOWN$TIME", "LNM$SYSTEM_TABLE") .nes. "" then deassign/system/exec shutdown$time
$ if f$trlnm("OPC$UNLOAD") .nes. "" then deassign opc$unload
$ if f$trlnm("OPC$REBOOT") .nes. "" then deassign opc$reboot
$ set uic 'saved uic
$ saved_privs = f$setprv(saved_privs)
$ set message 'saved_msg'
$ exit %x10360001

```



0232 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

AUTOGEN  
LIS

NETCONFIG  
LIS

STARTUP  
LIS

VMSINSTAL  
MEM

SWAPFILES  
LIS

TRANSERR  
LIS

MAKEROOT  
LIS

SHUTDOWN  
LIS

SPKITBLO  
LIS

VMSINSTAL  
LIS

VMSUPDATE  
LIS